

姚攀的博客 1.01^365=31.78

Engineers are versatile minds who create links between science, technology, and society.

- ≡ 目录视图

≡ 摘要视图

RSS 订阅

«从Lucene到Elasticsearch：全文检索实战»

希望本书能对您的工作和学习带来帮助，感谢支持！



当当 京东 亚马逊 天猫

Lucene、ES、ELK开发交流群:370734940

 加入QQ群

个人资料



yyyseb

关注

发私信



访问： 844555次

积分： 8642

等级： 

排名： 第2540名

图灵赠书——程序员11月书单 【思考】Python这么厉害的原因竟然是！ 感恩节赠书：《深度学习》等异步社区优秀图书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈

Spring全家桶(八)AOP核心思想与AspectJ 5种类型通知

标签： [spring](#) [aop](#) [Aspectj](#) [Aspect](#)

2017-05-17 00:02 1059人阅读 评论(0)

≡ 分类：

[spring \(8\)](#)

！ 版权声明：本文为博主原创文章,未经博主允许禁止转载(<http://blog.csdn.net/napoay>)

目录(?) [\[+\]](#)

一、AOP核心思想

AOP是Aspect-Oriented Programming的缩写，翻译为面向切面编程。我个人理解切面就是一个方面。

例子，一个接口里面有增删改查四个方法：

```
1 package com.stuspring.aop.impl;
2
3 /**
4  * Created by bee on 17/5/15.
5  */
6 public interface ArithmeticCalculator {
7     int add(int i,int j);
8     int sub(int i,int j);
9     int mul(int i,int j);
10    int div(int i,int j);
11 }
```

实现类：

```
1 package com.stuspring.aop.impl;
2
3 import org.springframework.stereotype.Component;
4
5 /**
6  * Created by bee on 17/5/15.
7  */
8 @Component("arithmeticCalculator")
```

原创： 202篇	转载： 2篇
译文： 6篇	评论： 451条

我的课程

Elasticsearch 5.4新闻搜索项目实战

StackOverFlow

<http://stackoverflow.com/users/6526424>
统计

阅读排行

[搜索]ElasticSearch Java Api(...	(61756)
Elasticsearch索引mapping的写...	(36806)
Elasticsearch 5.X下JAVA API使...	(31047)
搭建Elasticsearch 5.4分布式集群	(29631)
Elasticsearch 5.1.1 head插件安...	(25822)
Elasticsearch java api(五) Bulk...	(21822)
ElasticSearch Java Api(四)-删...	(19887)
mac命令行启动tomcat	(18615)
Elasticsearch 5 Ik+pinyin分词...	(17900)
Elasticsearch shield权限管理详解	(17369)

博客专栏



Spring全家桶

文章： 10篇

阅读： 10622



MapReduce编程

文章： 7篇

阅读： 24325



Lucene&Elasticsearch&ELK 专栏

文章： 42篇

阅读： 461286



数据库

文章： 9篇

阅读： 18151



J2EE

文章： 24篇

阅读： 92799



ruby on rails

文章： 18篇

阅读： 37381

数据结构与算法

```
9 public class ArithmeticCalculatorImpl implements ArithmeticCalculator {
10     @Override
11     public int add(int i, int j) {
12         int result=i+j;
13         return result;
14     }
15
16     @Override
17     public int sub(int i, int j) {
18         int result=i-j;
19         return result;
20     }
21
22     @Override
23     public int mul(int i, int j) {
24         int result=i*j;
25         return result;
26     }
27
28     @Override
29     public int div(int i, int j) {
30         int result=i/j;
31         return result;
32     }
33 }
```

如果想给这四个方法分别加上前置日志功能，以其中一个为例,add方法变这样：

```
1     @Override
2     public int add(int i, int j) {
3         System.out.println("The method add begins with " +i+","+j);
4         int result=i+j;
5         return result;
6     }
```

手动给每个方法都加上固然可行，但是维护起来过于麻烦，面向切面编程就是为了解决这一问题。AOP的好处就是每个逻辑位于一个位置，代码不分散便于维护和升级，业务模块更简洁。

加一个日志切面：

```
1 package com.stuspring.aop.impl;
2
3 import org.aspectj.lang.JoinPoint;
4 import org.aspectj.lang.annotation.After;
5 import org.aspectj.lang.annotation.Aspect;
6 import org.aspectj.lang.annotation.Before;
7 import org.springframework.stereotype.Component;
8
9 import java.util.Arrays;
10 import java.util.List;
11
12 /**
13  * Created by bee on 17/5/15.
14  */
```



文章：13篇

阅读：17571

文章分类

Elasticsearch (43)

Lucene (15)

hadoop (16)

J2EE (35)

数据结构 (16)

前端 (16)

ruby (18)

机器学习 (4)

python (5)

linux (11)

latex (2)

数据库 (12)

spring (9)

tools (1)

hibernate (1)

mybatis (1)

生活 (1)

文章存档

2017年12月 (9)

2017年11月 (1)

2017年10月 (3)

2017年09月 (2)

2017年08月 (1)

展开

最新评论

MapReduce编程(六) 从HDFS导入数据到Elasticsearch...
qq_38094271 : 用最新版本的那个 成功了。
注意json 文件最后一行要换行

《从Lucene到Elasticsearch:全文检索实战...
牙小木 : 书已买, 正在看

Lucene扩展停用词字典与自定义词库
yyseb : @cuipeng6561:这个文件在ik的目录下。

Lucene扩展停用词字典与自定义词库
cuipeng6561 : IKAnalyzer.cfg.xml 这个文件是要自己写么?

新人千万不要在 Windows 上使用 Ruby on Rails...
韩大喵 : 对于新手的我来说的, 相当中肯!

《从Lucene到Elasticsearch:全文检索实战...
艾鹤 : 6666, 恭喜发财。

Elasticsearch 5.X下JAVA API使用指南
qq_41296216 : @napoay:maven坐标是 <code>org.springframework</code>...
<code>org.springframework</code>...

Elasticsearch java api(五) Bulk批量索引
yyseb : @wangmaohong0717:可以不指定, 会自动生成。

Elasticsearch 5.X下JAVA API使用指南
yyseb : @qq_41296216:贴一下maven坐标吧, 或者, 博客上有QQ群, 加群讨论

```
15
16 @Aspect
17 @Component
18 public class LoggingAspect {
19
20     /**
21      * 前置通知 方法开始之前执行
22      * @param joinPoint
23      */
24     @Before("execution(public int com.stuspring.aop.impl.ArithmeticCalculatorImpl.*(..))")
25     public void beforeMethod(JoinPoint joinPoint) {
26         String methodName = joinPoint.getSignature().getName();
27         List<Object> args = Arrays.asList(joinPoint.getArgs());
28         System.out.println("The method " + methodName + " begins");
29     }
30
31     /**
32      * 后置通知, 方法执行完之后执行, 不论方法是否出现异常
33      * 后置通知中不能访问目标方法的执行结果
34      */
35     @After("execution(public int com.stuspring.aop.impl.ArithmeticCalculatorImpl.*(..))")
36     public void afterMethod(JoinPoint joinPoint) {
37         String methodName = joinPoint.getSignature().getName();
38         List<Object> args = Arrays.asList(joinPoint.getArgs());
39         System.out.println("The method " + methodName + " ends with " + args);
40     }
41 }
```

新建spring配置文件beans-aspect.xml:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:aop="http://www.springframework.org/schema/aop"
5       xmlns:context="http://www.springframework.org/schema/context"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7         http://www.springframework.org/schema/beans/spring-beans.xsd
8         http://www.springframework.org/schema/context
9         http://www.springframework.org/schema/context/spring-context-4.3.xsd
10        http://www.springframework.org/schema/aop
11        http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
12
13     <context:component-scan base-package="com.stuspring.aop.impl"/>
14     <aop:aspectj-autoproxy/>
15 </beans>
```

附maven依赖:

```
1 <dependency>
2     <groupId>org.springframework</groupId>
3     <artifactId>spring-context</artifactId>
4     <version>4.3.8.RELEASE</version>
5 </dependency>
6
```

Elasticsearch java api(五) Bulk批量索引
hello-friend : 批量索引数据的时候, 必须指定_id吗

```
7      <dependency>
8          <groupId>org.springframework</groupId>
9          <artifactId>spring-webmvc</artifactId>
10         <version>4.3.8.RELEASE</version>
11     </dependency>
12
13     <dependency>
14         <groupId>org.springframework</groupId>
15         <artifactId>spring-aop</artifactId>
16         <version>4.3.8.RELEASE</version>
17     </dependency>
18     <dependency>
19         <groupId>org.springframework</groupId>
20         <artifactId>spring-aspects</artifactId>
21         <version>4.3.8.RELEASE</version>
22     </dependency>
23
24     <dependency>
25         <groupId>org.aspectj</groupId>
26         <artifactId>aspectjrt</artifactId>
27         <version>1.8.10</version>
28     </dependency>
29
30     <dependency>
31         <groupId>org.aspectj</groupId>
32         <artifactId>aspectjweaver</artifactId>
33         <version>1.8.10</version>
34     </dependency>
```

二、五种类型通知

1. @Before前置通知: 方法开始之前执行。

```
1  /**
2   * 前置通知 方法开始之前执行
3   * @param joinPoint
4   */
5   @Before("execution(public int com.stuspring.aop.impl.ArithmeticCalcul
6   public void beforeMethod(JoinPoint joinPoint) {
7       String methodName = joinPoint.getSignature().getName();
8       List<Object> args = Arrays.asList(joinPoint.getArgs());
9       System.out.println("The method " + methodName + " begins with " +
10  }
```

2. @After后置通知: 方法开始之后执行, 不论方法是否出现异常。

```
1  /**
2   * 后置通知, 方法执行完之后执行, 不论方法是否出现异常
3   * 后置通知中不能访问目标方法的执行结果
4   */
5   @After("execution(public int com.stuspring.aop.impl.ArithmeticCalcula
6   public void afterMethod(JoinPoint joinPoint) {
```

```

7         String methodName = joinPoint.getSignature().getName();
8         List<Object> args = Arrays.asList(joinPoint.getArgs());
9         System.out.println("The method " + methodName + " ends with " + a
10    }

```

3. @AfterReturning: 返回通知, 在方法返回结果之后执行。

```

1    /**
2     * 返回通知, 在方法正常结束之后执行的代码
3     * 返回通知可以访问方法的返回值
4     */
5
6     @AfterReturning(value="execution(public int com.stuspring.aop.impl.ArithmeticCalculation.calculate())",
7     public void afterReturning(JoinPoint joinPoint, Object result) {
8
9         String methodName=joinPoint.getSignature().getName();
10        System.out.println("The method "+methodName+" ends with "+result)
11    }

```

4. @AfterThrowing: 异常通知, 在方法抛出异常之后。

```

1    /**
2     * 异常通知:在方法抛出异常之后执行
3     * @param joinPoint
4     * @param e
5     */
6
7     @AfterThrowing(value="execution(public int com.stuspring.aop.impl.ArithmeticCalculation.calculate())",
8     public void afterThrowing(JoinPoint joinPoint, Exception e){
9         String methodName=joinPoint.getSignature().getName();
10        System.out.println("The method "+methodName+" occurs execution: "+e.getMessage());
11    }

```

5. @Around: 环绕通知, 围绕方法执行。

```

1    /**
2     * 环绕通知需要携带ProceedingJoinPoint类型的参数
3     * 环绕通知类似于动态代理的全过程: ProceedingJoinPoint类型的参数可以决定是否执行目标方法
4     * 环绕通知必须有返回值, 返回值即为目标方法的返回值
5     *
6     * @param pjd
7     */
8     @Around("execution(public int com.stuspring.aop.impl.ArithmeticCalculation.calculate())")
9     public Object aroundMethod(ProceedingJoinPoint pjd) {
10
11        Object result = null;
12        String methodName = pjd.getSignature().getName();
13
14        try {
15            //前置通知
16            System.out.println("--->The method " + methodName + " begins ");
17            //执行目标方法
18            result = pjd.proceed();

```

```

19         //后置通知
20         System.out.println("--->The method " + methodName + " ends wi
21
22     } catch (Throwable e) {
23         //异常通知
24         System.out.println("The method "+methodName+" occurs exceptio
25     }
26
27     //后置通知
28     System.out.println("The Method "+methodName+" ends!");
29     return result;
30 }

```

三、指定切面的优先级

切面的优先级可以用@Order注解指定，传入的整数值越小，优先级越高。

```

@Order(1)
@Aspect
@Component
public class LoggingAspect {

```

四、复用切点表达式

```

1  /**
2   * 定义一个方法用于声明切入点表达式。一般地，该方法不需要再写其它代码。
3   */
4   @Pointcut("execution(public int com.stuspring.aop.impl.ArithmeticCalculator
5   public void declareJoinPointExpression(){
6   }
7   /**
8   * 前置通知 方法开始之前执行
9   * @param joinPoint
10  */
11
12  @Before("declareJoinPointExpression()")
13  public void beforeMethod(JoinPoint joinPoint) {
14      String methodName = joinPoint.getSignature().getName();
15      List<Object> agrs = Arrays.asList(joinPoint.getArgs());
16      System.out.println("The method " + methodName + " begins with " + agrs);
17  }

```

外部类引用可以使用报名加方法名的方法。

五、配置文件方式配置AOP

接口：

```
1 package com.stuspring.aop.fileimpl;
2
3 /**
4  * Created by bee on 17/5/16.
5  */
6 public interface ArithmeticCalculator {
7     int add(int i,int j);
8     int sub(int i,int j);
9     int mul(int i,int j);
10    int div(int i,int j);
11 }
```

实现类:

```
1 package com.stuspring.aop.fileimpl;
2
3 /**
4  * Created by bee on 17/5/16.
5  */
6 public class ArithmeticCalculatorImpl implements ArithmeticCalculator {
7     @Override
8     public int add(int i, int j) {
9         int result=i+j;
10        return result;
11    }
12
13    @Override
14    public int sub(int i, int j) {
15        int result=i-j;
16        return result;
17    }
18
19    @Override
20    public int mul(int i, int j) {
21        int result=i*j;
22        return result;
23    }
24
25    @Override
26    public int div(int i, int j) {
27        int result=i/j;
28        return result;
29    }
30 }
```

日志切面:

```
1 package com.stuspring.aop.fileimpl;
2
3 import org.aspectj.lang.JoinPoint;
4
5 /**
6  * Created by bee on 17/5/16.
7  */
```

```
8 public class LoggingAspect {
9
10     public void beforeMethod(JoinPoint joinPoint){
11         String methodName=joinPoint.getSignature().getName();
12         System.out.println("The method begins with "+methodName);
13     }
14 }
```

参数验证切面:

```
1 package com.stuspring.aop.fileimpl;
2
3 import org.aspectj.lang.JoinPoint;
4
5 /**
6  * Created by bee on 17/5/16.
7  */
8 public class ValidationAspect {
9
10     public void validateArgs(JoinPoint joinPoint){
11         System.out.println("validationMethod.....");
12     }
13 }
```

Spring配置文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:aop="http://www.springframework.org/schema/aop"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6       http://www.springframework.org/schema/beans/spring-beans.xsd
7       http://www.springframework.org/schema/aop
8       http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
9
10     <!--配置bean-->
11     <bean id="arithmeticCalculator" class="com.stuspring.aop.fileimpl.Arithmeti
12     <bean id="loggingAspect" class="com.stuspring.aop.fileimpl.LoggingAspect"/>
13     <bean id="validationAspect" class="com.stuspring.aop.fileimpl.ValidationAs
14
15     <aop:config>
16         <!--配置切面表达式-->
17         <aop:pointcut id="pointcut" expression="execution(public int com.stuspr
18         <!--配置切面通知-->
19         <aop:aspect ref="loggingAspect" order="2">
20             <aop:before method="beforeMethod" pointcut-ref="pointcut"/>
21         </aop:aspect>
22
23         <aop:aspect ref="validationAspect" order="1">
24             <aop:before method="validateArgs" pointcut-ref="pointcut"/>
25         </aop:aspect>
26
27     </aop:config>
28 </beans>
```


Main测试方法:

```
1 package com.stuspring.aop.fileimpl;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 /**
7  * Created by bee on 17/5/16.
8  */
9 public class Main {
10
11     public static void main(String[] args) {
12         ApplicationContext ctx=new ClassPathXmlApplicationConte
13         ArithmeticCalculator calculator= (ArithmeticCalculator)
14         System.out.println(calculator.add(2,4));
15     }
16 }
```

顶

2

踩

0

- [上一篇](#) Spring全家桶(七)通过注解配置Bean
- [下一篇](#) Spring全家桶(九)Spring JdbcTemplate

相关文章推荐

- 比较分析 Spring AOP 和 AspectJ 之间的差别
 - MySQL在微信支付下的高可用运营--莫晓东
 - 关于 Spring AOP (AspectJ) 你该知晓的一切
 - 容器技术在58同城的实践--姚远
 - AOP之@AspectJ技术原理详解
 - SDCC 2017之容器技术实战线上峰会
 - Spring 之AOP AspectJ切入点语法详解（最全面、...
 - SDCC 2017之数据库技术实战线上峰会
- Spring详解-----基于@ASpectJ的AOP
 - 腾讯云容器服务架构实现介绍--董晓杰
 - SpringAop与AspectJ的联系与区别
 - 微博热点事件背后的数据库运维心得--张冬洪
 - spring学习笔记（3）-aspectj的五种通知方法
 - Spring三大核心思想之三：AOP
 - Spring AOP 实现和一些核心思想
 - Spring IoC与AOP的核心思想（转载）

查看评论

暂无评论

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 